Efficient Optical Flow and Stereo Vision for Velocity Estimation and Obstacle Avoidance on an Autonomous Pocket Drone

Kimberly McGuire[1], Guido de Croon[1], Christophe De Wagter[1], Karl Tuyls[2] and Hilbert Kappen[3]

***Abstract*—Micro Aerial Vehicles (MAVs) are very suitable for flying in indoor environments, but autonomous navigation is challenging due to their strict hardware limitations. This paper presents a highly efficient computer vision algorithm called Edge-FS for the determination of velocity and depth. It runs at 20 Hz on a 4 g stereo camera with an embedded STM32F4 microprocessor (168 MHz, 192 kB) and uses edge distributions to calculate optical flow and stereo disparity. The stereo-based distance estimates are used to scale the optical flow in order to retrieve the drone's velocity. The velocity and depth measurements are used for fully autonomous flight of a 40 g pocket drone only relying on on-board sensors. This method allows the MAV to control its velocity and avoid obstacles.**

***Index Terms*—Aerial Systems: Perception and Autonomy, Autonomous Vehicle Navigation, Micro/Nano Robots, Visual-Based Navigation**

I. Introduction

DEPLOYMENT of Micro Aerial Vehicles (MAVs) is important for indoor tasks such as inspections, search-and-rescue operations, green house observations and more. Tiny MAVs, also called pocket drones (<50 g, as in Fig. 1), are ideal for maneuvering through very narrow spaces, as often occurs in indoor environments. In order for them to autonomously navigate through a GPS-deprived area, there are several on-board sensors to consider (laser rangers, motion sensors, infrared rangers, sonar). The pocket drone's sensor of choice is a RGB camera. It is the most energy efficient and versatile sensing option, as multiple variables can be observed from the image stream: obstacles, motion, object recognition and more.

Using cameras enables the Micro Aerial Vehicle (MAV) to extract essential information for autonomous navigation. A stereo vision setup with two cameras has been particularly successful, for instance for obstacle avoidance [1]. Since there are strict limitations on energy expenditure, sensing, and processing capabilities on a pocket drone, even relatively efficient stereo vision methods [2][3] are computationally too heavy to run on-board a microprocessor. Therefore, an even more efficient stereo vision algorithm was developed, which is able to run at 10 Hz on a 20 g flapping wing MAV, the DelFly Explorer [4]. It is still the lightest fully autonomous MAV to this date, which can fly through a room and avoid obstacles with purely onboard sensing and processing [5].

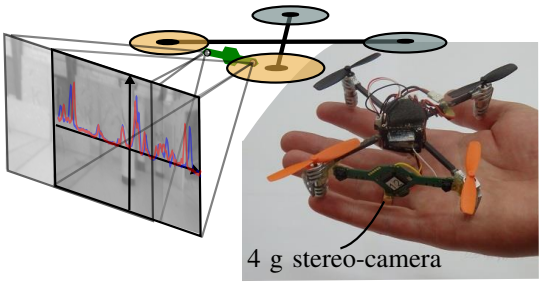

Fig. 1. Pocket drone with a lightweight forward looking 4g stereo camera. A very efficient vision algorithm runs embedded on the STM32F4 processor (168 MHz, 192 kB), to determine velocity and depth necessary for the pocket drone's visual navigation.

Since tailed flapping wing MAVs such as the DelFly Explorer are passively stable, there is no need to compute their velocity to compensate for drift. However, for inherently unstable platforms like a quadcopter, velocity estimation is necessary for stabilization when navigating in constrained areas. Optical flow is the way in which objects move in two sequential images and is the most important visual cue for velocity estimation. It can be calculated in a dense manner (Horn-Schunck [6], Färneback [7]) or a sparse manner, e.g., by tracking features such as Shi-Tomasi [8] or FAST [9] over time with a Lucas-Kanade tracker [10]. These types of techniques have proven themselves on numerous occasions [11], nonetheless do require a platform with a decent amount of computing power. On a pocket drone such standard optical flow methods either cannot be run in real-time or take consist of an unpractically large part of the processing time, leaving little to no room for other types of processing. Especially when autonomous flight is the final goal, optical flow determination will only constitute a part of what the MAV has to do, as much more information can be retrieved from the image stream.

In order to design a computationally much more efficient optical flow algorithm, we have drawn inspiration from the study in [12], which proposed using spatial edge distributions to track motion in the image. Specifically, in [13], we presented EdgeFlow, which improved upon the work in [12] by introducing a variable time horizon for determining sub-pixel flow. EdgeFlow ran embedded at 30 Hz on a lightweight stereo camera positioned underneath a pocket drone. The stereo camera was pointing down and was estimating optical flow and a global height estimate, assuming that it was looking at a flat ground surface. With these, the MAV determined its own velocity and used this in a guided control, where it autonomously matched externally-given velocity references. However, a 4 g stereo camera for a 40 g pocket drone is

Manuscript received: October 09, 2016; Revised December 07, 2017; Accepted January 07, 2017.

This paper was recommended for publication by Editor Editor name upon evaluation of the Associate Editor and Reviewers' comments.

[1] Faculty of Aerospace Engineering, Delft University of Technology, NL k.n.mcguire@tudelft.nl, g.c.h.e.decroon@tudelft.nl

[2] Department of Computer Science, University of Liverpool, UK

[3] Faculty of Science, Radboud University of Nijmegen, NL

Digital Object Identifier (DOI): see top of this page.

significant, so it is a waste to have this "heavy" sensor looking downward and not using it to avoid obstacles in the flight direction.

This paper presents a major extension of EdgeFlow, which enables the stereo camera to face forward on a MAV, so it can be used for navigation purposes. As the pocket drone will now be facing hallways, rooms, doors etc., the assumption of looking straight at a flat plane will not hold anymore. The same matching paradigm used to determine EdgeFlow, will now be used to not only calculate optical flow but also stereo depth over the entire image. *EdgeStereo*, as called for convenience, uses the so-determined distances to properly scale the locally observed optical flow in order to retrieve a velocity estimate. This combination of EdgeFlow and EdgeStereo will be called *Edge-FS*.

Our main contribution is that the presented method provides both velocity and distance estimates, while still being computationally efficient enough to run close to the frame rate on a very limited embedded processor. As such, the method enables unstable MAVs such as tiny quadcopters to perform fully autonomous flights in unknown environments. The EdgeFlow and EdgeStereo methods will be explained in more detail in section II. Off-line results for velocity estimates with a set of images is shown in section III. From here, the algorithm is embedded on the lightweight stereo camera and placed on 40 g pocket drone for velocity estimation (section IV-B). Finally, the velocity estimate is used together with EdgeStereo-based obstacle detection to perform fully autonomous navigation in an environment with obstacles (section IV-C). This is followed by some concluding remarks.

A. Related Work

In related research, several works have achieved optical flow based control of a MAV, e.g., [14][15][16]. As mentioned in the introduction, the standard optical flow methods are computationally too heavy to run on a quadcopter of less than 50 g. For instance, Dunkley et al. have flown with a 25 g quadcopter before, while computing optical flow for visual odometry [17]. However, this was done on an external computer. As miniaturization of hardware also poses a limitation on communication bandwidth, this can result in a significant delay in the controls. To obtain full autonomy, it would be wise to uncouple a MAV of any external dependencies.

To design extremely lightweight MAVs for autonomous flight, some researchers looked into EMD sensors [18] and other 1D signal sensors [19]. Briod et al. [20] proposed the design of a 45 g quadcopter for optical flow based control with 1D flow sensors. They followed up with this research on a heavier 278 g platform containing 8 of these sensors pointing in all directions [21]. With this they could hover the quadcopter in various cluttered environments. The results are impressive, nevertheless they were achieved by using multiple single purpose sensors. As they can only sense motion, it does not leave much room to detect other variables necessary for navigation.

More similar to our research, Moore et al. implemented an efficient optic flow algorithm on a small lightweight (2 g) omnidirectional camera system on a 30 g helicopter [22]. With a ring of 8 low-resolution image chips (64 x 64 pixels), the MAV could compute optical flow. It did this by computing the edges, compressing the images and calculate the displacement by block matching which resulted in translational optical flow. The vision calculations where done on-board the helicopter with 10 Hz, yet the flight controls where computed off-board. Although the potential of a full on-board implementation is there, the redundancy lies in the ratio of cameras to sensed variables. One camera has the potential of detecting flow in 3 directions; they used 8 to only detect 2 (forward and sideways velocity).

Optical flow can also be used to detect obstacles [23], however the MAV needs to be constantly on the move. This is not required if stereo vision is used for depth information. With this, Oleynikova et al. developed a reactive avoidance controller for a quadcopter (30 cm in diameter) [24]. From the obtained stereo disparity map, they accumulated the values along the columns to get a summed disparity factor. Assuming that the obstacles are vertical and long, these can be detected quickly. The stereo map was calculated over the entire image first before accumulation to a vector. This significantly impacts the amount of computation making it less suitable for implementation on a smaller MAV.

II. Velocity and Depth from Edges

To achieve autonomous navigation with a camera on an unstable pocket drone, we need to obtain two variables: velocity and depth. In the introduction we mentioned that many of the mainstream computer vision will be computationally too heavy to run on the pocket drone. In [13], we presented EdgeFlow, which can detect optical flow within the image in a semi-dense but computationally efficient manner, embedded on a 4 g stereo board. During the experiments, the stereo camera looked down to the ground, estimating the pocket drone's forward and sideways velocity. This section will explain the modifications that are necessary to make the stereo camera point forward and still be able to measure those variables. EdgeFlow will be concisely recapped. Subsequently, we will present its extension with EdgeStereo to Edge-FS, which will be used for obstacle detection in the experiment part of this paper.

A. From Camera to State

When looking orthogonally at a planar ground surface while moving, the optical flow field is rather simple and allows for easy determination of the forward and sideways velocities with the help of a single height measurement. But to navigate without bumping into anything, the MAV needs to see objects in the direction of motion, which in this study is forward. Due to the likely non-planar (3D structure) of the environment in forward direction, the optical flow field will become more complex. Moreover, the forward velocity now can only be observed by means of the divergence of optical flow, which is more difficult to determine, especially close to the focus of expansion. Here we delve into how we determine the velocities with the help of forward facing stereo images. In principle, the unscaled velocities and the rotation rates can

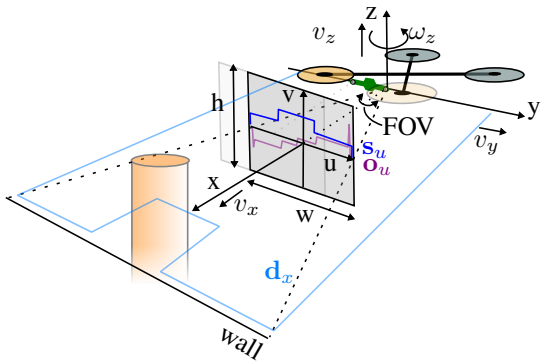

Fig. 2. The MAV's body fixed coordinates with respects to the camera axis, shown for the left camera (XYZ). The conventional aircraft coordinates of east north up is used for the MAV as the camera. The image coordinates in width and height are represented as u and v respectively.

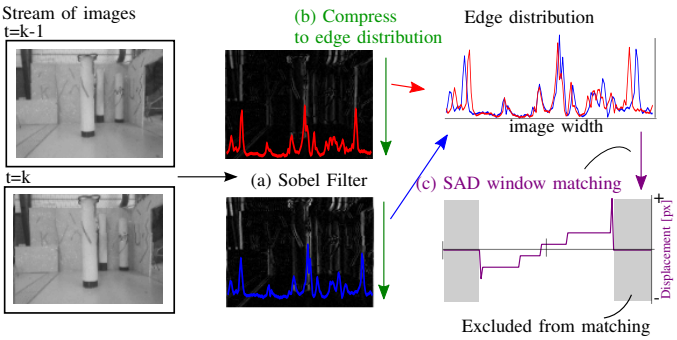

Fig. 3. The matching algorithm for both EdgeFlow as EdgeStereo. The images' gradients (a) calculated by a Sobel filter, (b) summed up to an edge distribution. These are (c) matched with other edge distribution. The gray areas are excluded sections (equal to the range plus half SAD block size).

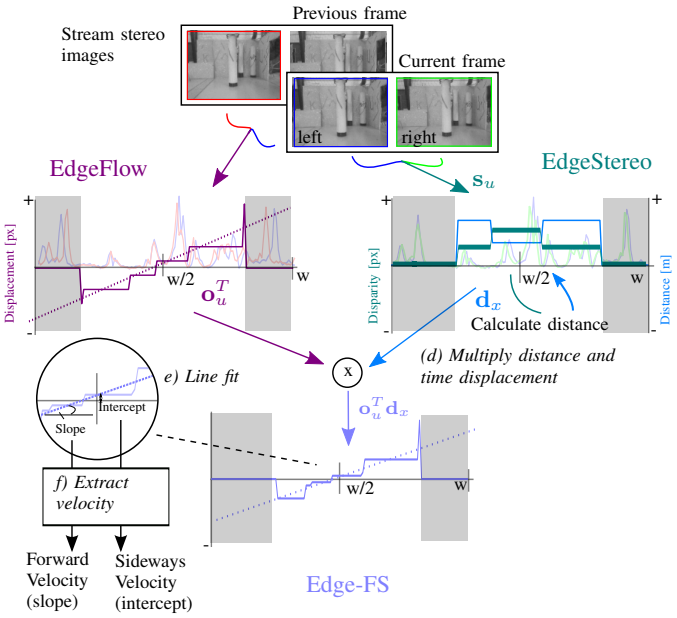

Fig. 4. The temporal pixel disparities per column of EdgeFlow is (d) scaled by EdgeStereo. Following (4), e) a line fit is done on this array of values, from which the forward and sideways velocities can be extracted from the slope and intercept, respectively.

be determined from the image alone, according to the paper of [25]. Longuet-Higgins and Prazdny implied that measured flow ($\mathbf{o}_u$) is the summation of a translational flow ($\mathbf{o}_u^T$) and rotational component ($\mathbf{o}_u^R$). Before estimating the horizontal planar velocity, we first have to determine $\mathbf{o}_u^R$.

Although [25] assumes rotations in all directions, we can make simpler assumptions for the pocket drone. Fig. 2 shows the placement and axis definition of the drone and camera. For obstacle avoidance it is essential to look in the direction of motion, which in this case is the direction of the positive x axis. Here, correctional pitch and roll motion for drift compensation will be relatively small, but yaw rotations will be more common. Assuming that the latter only has significant effect on the optical flow, $\mathbf{o}_u^R$ can be approximated (assuming small angles) using the gyroscopes on the on-board IMU of the pocket drone:

$$o_{u,i}^R \approx \omega_Z \cdot \frac{w}{\alpha_{FOV}} \tag{1}$$

$$\mathbf{o}_u^R = [o_{u,1}^R, \ldots, o_{u,w}^R] \tag{2}$$

where w is the width of the image, α_{FOV} is the angle of the Field of View (FOV) and ω_Z is the yaw rotation measured from the gyroscopes.

Now that $\mathbf{o}_u^R$ is known, we can isolate $\mathbf{o}_u^T$ to determine the pocket drone's forward (v_x) and sideways velocity. With the coordinate system we use in this paper (Fig. 2), Longuet's equation of $\mathbf{o}_u^T$ is expressed as:

$$\mathbf{o}_u^T = (-v_y + \mathbf{x}v_x)/\mathbf{d}_x \tag{3}$$

$$\mathbf{d}_x\mathbf{o}_u^T = -v_y + \mathbf{x}v_x \tag{4}$$

Where $\mathbf{x}$ is an array of indices of the image columns. Depth, $\mathbf{d}_x$, scales the optical flow resulting in motion parallax, as close objects appear to move faster than objects far away. In [13], a global height estimate was used to scale the optical flow back to velocities, which is sufficient if the camera is looking at a flat floor or perpendicular to a straight wall. This assumption will not hold when the MAV is flying towards a wall at an angle or whenever obstacles at different distances are in the field of view. This non-constant depth needs to be accounted for when scaling the optical flow, therefore the stereo depth is needed over the entire size of the image for a better velocity estimate. Local right-left image disparity from a stereo camera can be transformed to actual depth in meters by using the camera parameters, with the following approximation:

$$\mathbf{d}_x \approx \frac{w \cdot r}{\alpha_{FOV} \cdot \mathbf{s}_u} \tag{5}$$

where r is the baseline between the two cameras, and the stereo disparities in pixels along the image columns is $\mathbf{s}_u$.

With depth $\mathbf{d}_x$ and translational optical flow $\mathbf{o}_u^T$, it is now possible to calculate the MAV's sideways and forward velocity by fitting a linear model to (4). In the next section we will explain how to obtain both optical flow and stereo depth from a stream of low resolution stereo images.

B. Procedure for Edge-FS

The matching principle of EdgeFlow is shown in Fig. 3. The images A and B are first filtered with a Sobel filter to get the horizontal gradients (Fig. 3(a)). These gradients are summed along the rows (compressed) to a (spatial) edge distribution

(Fig. 3(b)). From image A and B, the edge distributions are compared with Sum of Absolute Differences (SAD) block matching (Fig. 3(c)). This locates the similar patches of the edge distributions within a certain distance from each other, to obtain the pixel displacement between the two.

If image A and B from Fig. 3 are two temporal sequential images (t with $t-1$) in time, this will result in the pixel flow, thus EdgeFlow as presented in [13]. Based on the previous flow value, EdgeFlow adaptively chooses how far in time ($t-n$) it will compare the current edge distribution to. On top of that, the flow shift predicted by a yaw rotation ($\mathbf{o}_u^R$, as calculated in (2) will shift the start of the block matching scheme. This and the adaptive time horizon will be present for the experiments in this paper. Note that in [13], also the direction along the image height was used to estimate the forward velocity (as the camera was looking down). In this paper, it will not be used as the forward velocity (v_x) will now be subtracted from the divergence of Edge-FS.

Previously in [13], the entire edge distribution of the left and right image were matched to obtain a global depth estimate. To get a better velocity estimate with a forward camera, we need to use pixel disparity per column. To calculate both column-wise optical flow and stereo vision and keep the algorithm computationally efficient, the exact same matching principle of EdgeFlow (Fig. 3) is used, resulting in EdgeStereo. Disparity to depth in meters is calculated with the known camera parameters and (5) from the last section. Sequentially, EdgeStereo scales EdgeFlow to compensate for the motion parallax (see Fig. 4(d)), which results in the left side of (4). These values will then be fitted to a linear model (Fig. 4(e)), which gives us the slope and intercept of the line. With the camera parameters, the forward and sideways velocities are estimated (Fig. 4(f)).

III. Off-line Vision Experiments

Before implementing the algorithm on the actual stereo board, EdgeFlow was run on a set of stereo-images in MATLAB (version R2015b on a Dell Latitude E7450, i7-5600U CPU @ 2.60GHz processor). Fig. 5(a) shows screen shots of the data set used in this section, where the camera moves towards obstacles at different distances. In Fig. 5(b), EdgeFlow scaled by EdgeStereo, now dubbed as Edge-FS, results in the velocity estimates.

Edge-FS is contrasted against the well-known optical flow method developed by Färneback [7], a dense optical flow method (Fig. 5). Although less used than a more conventional KLT-tracker [10], preliminary analyses indicated it to be more suited for the low-resolution images used here. With its default parameters set as in MATLAB R2015b, the sparse magenta line illustrates that the KLT-tracker indeed has difficulties with the low-quality, low-resolution images (128 x 96 pixels).

For Färneback, depth is determined by matching the stereo-images with each other and converting the resulting pixel disparity to a distance. To get velocity, the same line-fit is used as for EdgeFlow[1], but here the whole image is considered

[1]The EdgeFlow code as embedded on the stereo camera has a mean computation time for EdgeFlow is 0.00134 seconds (compiled for Linux) and for Färneback is 0.00466 seconds on the same stereo-image data set.

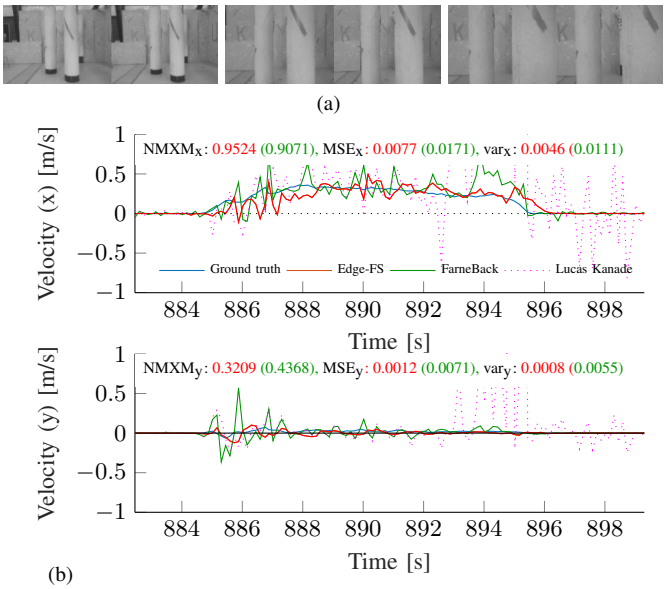

Fig. 5. (a) Several screen shots of the set of images used for off-line estimation of the velocity. Here the diversity in amount of texture can be seen. (b) Off-line velocity estimate calculated by Edge-FS and Färneback, held against the ground truth for the forward moving camera's data set.

rather than the compressed form like the edge distributions. After comparison of the methods with different parameters, both Edge-FS and Färneback are set up with a window size of 11 pixels and a search range of 15 pixels (Färneback's pyramid level at 1). Both forward (x) and sideways (y) velocity measurements, shown in Fig. 5, are compared against the "ground truth" as obtained with an OptiTrack motion capture system[2], with 24 infrared cameras. The plots also include several values to determine the quality of the velocity estimates: Mean Squared Error (MSE), Variance (VAR) and Normalized Maximum Cross-Correlation Magnitude (NMXM). A low MSE indicates greater similarity and low VAR is a smaller spread of the measurement from the ground truth. A high NMXM stands for a better shape correlation between the two. All these metrics indicate Edge-FS to obtain more accurate results on this data set than the computationally more expensive Färneback method.

It is important to note that because of the nonlinear relation between pixel disparity and depth in stereo vision, far distances are measured less accurately. The disparities for further distances will become sub-pixel and hard to determine. This is especially relevant to the small stereo board used in this study, which we set up to use 128 x 96 pixel images for the 57.4 x 44.5 deg FOV. Also, the translational optical flow of objects is harder to measure when they are further away, since it becomes sub-pixel as well. Hence, both terms on the left in (4), $\mathbf{s}_u$ and $\mathbf{d}_x$ become less accurate at far distances. This correlation between distance and accuracy can be seen in the box plot of Fig. 6.

Besides the difficulty with larger distances, which is fundamental to stereo vision, Edge-FS also has a bit difficulty determining the forward flow when there is a large lateral

[2]www.optitrack.com

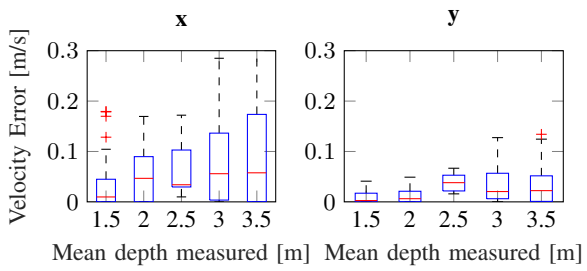

Fig. 6. Boxplot of the absolute velocity estimate error of Edge-FS, compared against the mean observed depth.

motion. Sideways velocity has a 0^{th} order effect on the flow, while forward velocity information is captured by the divergence of the flow field, which is a 1^{st} order effect. Therefore, the forward velocity is more subject to noise and harder to estimate (this can be observed in Fig. 6 as the errors are generally higher for the x-direction than the y-direction). In this work, the MAV will mostly fly forward. In this situation lateral flow is kept very small while the divergence is larger and more observable. A larger SAD window size and filtering are used to correct for the remaining noise. While further analysis of the noise is beyond the scope of this article, the reader is encouraged to look at it in more detail by using our MATLAB code for Edge-FS including a large diverse image data-set[3].

IV. Experiments on the Pocket Drone

In this section, we explain the implementation of Edge-FS on-board a pocket drone and how it is used in an autonomous obstacle avoidance task. We will first present the velocity estimates by Edge-FS during flight on the stereo board. Subsequently, a closed loop flight is shown, where the drone autonomously navigates through a room, while maintaining its velocity and avoiding obstacles.

A. Hardware specifics

Edge-FS local runs embedded on the stereo camera (as introduced in [4]). Fig. 7 displays two cameras with 1/6 inch image sensors, with a baseline of 6 cm and a Field of View (FOV) of 57.5^o x 44.5^o. The stereo camera has an embedded microprocessor, an STM32F4 with a speed of 168 MHz and 196 kB of memory in which the largest consecutive memory block spans 128 kB. The cameras are configured to output stereo-images with a size of 128 x 96 pixels to fit within memory and processing constraints. The maximum reachable frame rate of the stereo camera is 30 Hz, which is not much affected by the computation of Edge-FS (approx. 0.0175 sec).

For the experiments, a pocket drone is equipped with a single front-facing stereo camera. A frame of a Walkera QR LadyBug[4] is adopted as a base. An adapted smaller variant of the Lisa-MX[5] will be used the auto-pilot. The *Lisa-MXs* also carries an STM32F4 microprocessor, with a speed of 168 Hz

[3]https://github.com/knmcguire/EdgeFlow_matlab
[4]http://www.walkera.com/
[5]http://wiki.paparazziuav.org/wiki/Lisa/MX

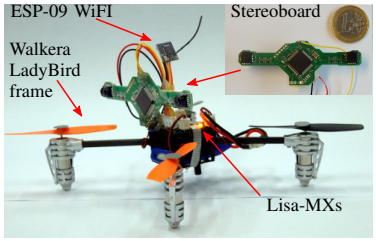

Fig. 7. The 4 g stereo camera mounted on the pocket drone.

and 1 MB of flash memory. With an ESP-09 WiFi module, telemetry can be broadcasted to the computer to receive all the measured variables required for validation. The entire assembly, including stereo camera and battery, weighs exactly 41.9 g.

The auto-pilot program flashed on the Lisa-MXs is Paparazzi[6]. The software runs entirely on-board the microprocessor which governs all the basic flight controls. An adaptive Incremental Nonlinear Dynamic Inversion (INDI) controller [26] is used for the attitude stabilization of the MAV. The guidance controller resides on top of the stabilization control, to calculate the desired pitch and roll angle, to achieve a desired altitude position or airspeed. In this paper, it will be applied to maintain a desired velocity. It will need the measurements from the stereo camera, operating in parallel with the Lisa-MXs.

B. Velocity Estimate

We have shown in section III that EdgeFlow can measure the camera velocity based on a collection of images. Now implemented in the 4 g stereoboard and fixed on a pocket drone, the question remains if it can still retain its quality with all the additional effects caused by motion and vibrations during flight.

Fig. 8(a), presents the velocity estimates of Edge-FS, during a manually controlled flight in front of a textured screen (screen shot in Fig. 8(b) and position in Fig. 8(c)). The same OptiTrack system used for the image data set (Fig. 5(a)), is monitoring its real velocity. The raw unfiltered velocity measurements of Edge-FS are contrasted with this ground truth with NMXM, VAR and MSE. Noticeable is that the forward velocity shows more noise peaks than the sideways velocity as expected (see Section III). However, in both directions, Edge-FS matches the ground truth adequately, which should be sufficient for the closed-loop flight.

To use the actual raw measurements in flight is undesirable. The most common way is to fuse these vision-based velocity estimates with the accelerometers. On a larger MAV than the pocket drone, this would be possible because of the damping. However, many vibrations are generated by the small propellers, which are in close proximity with the autopilot, the accelerometers readings contain too much noise. Therefore, in this paper, we only use a vision-only approach applying a median filter to the 5 last velocity measurements, to keep the delay to a minimum.

[6]http://wiki.paparazziuav.org/

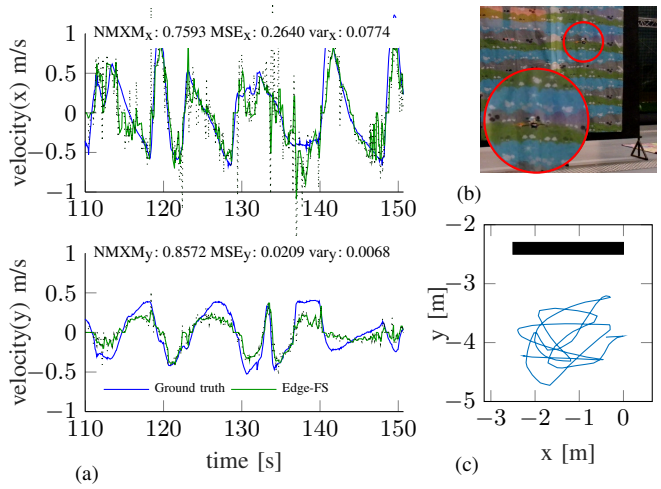

Fig. 8. (a) Velocity estimation by Edge-FS on the pocket drone, (b) a screen shot from the flight in front of a wall and (c) the position during a remote controlled maneuver. Dotted line is the unfiltered velocity estimate by Edge-FS.

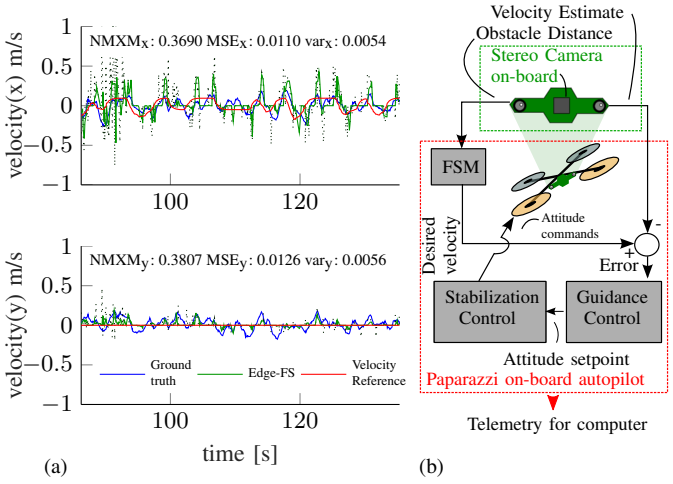

Fig. 9. (a) Velocity control on pocket drone with a wall force field and (b) the control scheme used, explaining the hierarchy of the on-board sensors and controllers.

C. Autonomous Obstacle Avoidance

In the previous subsection, we showed validation of the velocity estimate as calculated by Edge-FS. Now we will present a closed-loop flight, where the pocket drone avoids obstacles identified by means of its stereo vision, while guided by its velocity estimates. The main goal of this experiment is to show the potential of the proposed algorithms for full autonomous navigation. In this section, the vertical position as measured by OptiTrack is exclusively used for height control, as no position measurement is used in the horizontal plane (solely for validation afterwards). This is where the MAV uses its velocity estimates by Edge-FS.

Fig. 9(b) displays the basic control scheme for the navigation task. It determines a desired velocity to avoid collisions. The error between the estimate and the desired velocity is the input to the velocity guidance controller, which sets an attitude set-point for the stabilization. Subsequently, EdgeStereo determines the nearest object to camera. If too close, it will produce a backward velocity reference to the guidance controller (a force field), therefore preventing the pocket drone from hitting the wall face-on. Fig. 9(a) shows the readings from a short flight of a simple hover with the obstacle force field.

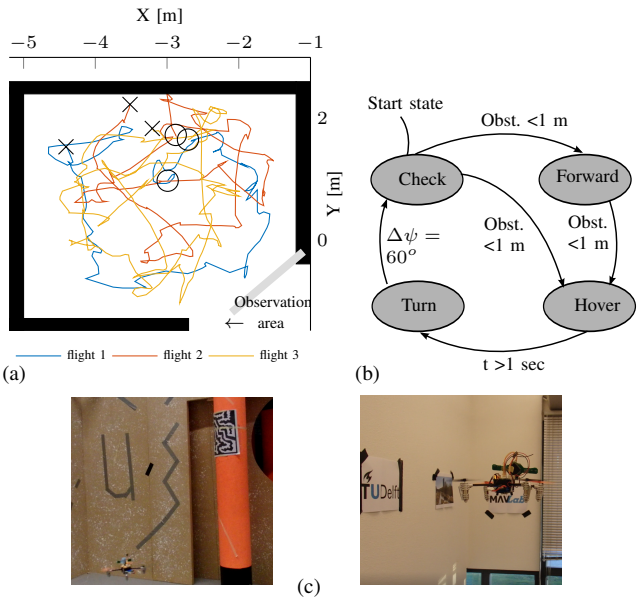

Fig. 10. (a) A position plot of 3 flights, from which the first lasted 91 seconds, the second 101 seconds and the third 122 seconds. (b) shows screen shots of the experiments in the flight arena (left) and in a real-world office (right). Some posters were added to the latter to provide extra texture.

When encountering a wall/obstacle, the pocket drone will need to move away from the situation. The avoidance scheme is a simple finite state machine (FSM) with 4 behavioral states (see Fig. 10(b)). It starts in *check* mode, where the pocket drone will check if there is a detected obstacle within 1 meter by EdgeStereo. If the way is clear, the pocket drone moves *forward* with a constant speed (set now to 0.3 m/s), guided by the velocity estimate from Edge-FS. If it detects an object on its path, the MAV will first hover for 1 second actively controlling the forward velocity to zero. Then it will turn quickly with a constant angle relative to the heading (here $\Delta\psi = 60^o$). Immediately thereafter, the MAV will evaluate the situation in the *check* mode and proceeds from there.

We conducted multiple autonomous flights with the pocket drone. Fig 10(a) shows the result of 3 representative flights of the pocket drone with the forward looking stereo camera. The pocket drone has to navigate in a small room of 4 x 4 meter with varying textured surfaces (screen shot of camera footage). All the flights lasted longer than 90 seconds, from which the longest duration was 122 seconds (flight 3). When the pocket drone brushed against the wall, the safety pilot took over the flight with a remote control for a safe landing. The most common failure case during the test flights, is that the MAV will approach the wall with a small angle. After the turn with constant angle, the drone will fly almost parallel to the wall which it can not detect due to its limited FOV. This is the case for flight 2 and 3, except for flight 1, in which case the pocket drone was facing the observer after a turn. Several flights of the pocket drone have been done within a real-world environment (Fig. 10(c)), which can be observed

with the accompanying video and YouTube list[7] in Fig. 10(c).

The mentioned failure case for the autonomous flights is difficult to overcome. If the MAV would turn and face a large open space, the distance for EdgeStereo could be far enough to compromise the quality for the velocity estimate due to the small base line of the stereo camera. As we already observed in Fig. 6, this would cause the pocket drone to drift, which is problematic when near a wall/obstacle after the turn. If an obstacle is not in its FOV, the chances of collision significantly increases. This could be solved by merging the *check* and *turn* node of the FSM, so it will only stop turning at a significant clear path. Another solution is to add a lightweight short range sensor on the sides of the pocket drone, so it will detect immediately if the drone is flying close and aside an obstacle.

The obstacle avoidance logic will need some additional work, however the experiments show that Edge-FS can be used in navigation overall. During the autonomous flight, the pocket drone was stabilizing itself using the velocity estimates of its forward camera alone.

V. Conclusion

A computationally efficient optical flow and stereo algorithm is presented in this paper, called Edge-FS. It runs embedded on a very lightweight stereo camera and can be carried by a 40 g pocket drone for determining velocity and depth. The presented algorithm allows the stereo camera to face forward, a direction in which a complex 3D structure can be expected.

We presented experiments where the pocket drone with the stereo camera autonomously navigated and avoided obstacles in an area of 4 x 4 meters. A simple finite state machine controller showed that the velocity estimates and the depth measurement can be used for fully autonomous flight. The current work lays the basis for stabilization and collision avoidance on pocket drones with a single, small stereo vision system.

Acknowledgments

We would like to show our gratitude to Kirk Scheper from the MAVlab, Delft University of Technology, for assistance in the development of the guided flight mode and the finite state machine. These results greatly improved the quality of the paper.

References

[1] X. Hu and P. Mordohai, "Evaluation of stereo confidence indoors and outdoors," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1466–1473.

[2] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[3] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 963–968.

[4] C. De Wagter, S. Tijmons, B. Remes, and G. de Croon, "Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4982–4987.

[5] S. Tijmons, G. de Croon, B. Remes, C. De Wagter, and M. Mulder, "Obstacle avoidance strategy using onboard stereo vision on a flapping wing mav," *arXiv preprint arXiv:1604.00833*, 2016.

[6] B. K. Horn and B. G. Schunck, "Determining optical flow," pp. 185–203, 1981.

[7] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*. Springer, 2003, pp. 363–370.

[8] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

[9] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 1508–1515.

[10] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, no. 1-10, p. 4, 2001.

[11] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1736–1741.

[12] D.-J. Lee, R. W. Beard, P. C. Merrell, and P. Zhan, "See and avoidance behaviors for autonomous navigation," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5609, pp. 23–34, 2004.

[13] K. McGuire, G. de Croon, C. de Wagter, B. Remes, K. Tuyls, and H. Kappen, "Local histogram matching for efficient optical flow computation applied to velocity estimation on pocket drones," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 3255–3260.

[14] V. Grabe, H. H. Bulthoff, D. Scaramuzza, and P. R. Giordano, "Nonlinear ego-motion estimation from optical flow for online control of a quadrotor UAV," *The International Journal of Robotics Research*, vol. 34, no. 8, pp. 1114–1135, 2015.

[15] H. Romero, S. Salazar, and R. Lozano, "Real-time stabilization of an eight-rotor UAV using optical flow," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 809–817, 2009.

[16] F. Kendoul, I. Fantoni, and K. Nonami, "Optic flow-based vision system for autonomous 3d localization and control of small aerial vehicles," *Robotics and Autonomous Systems*, vol. 57, no. 6, pp. 591–602, 2009.

[17] O. Dunkley, J. Engel, J. Sturm, and D. Cremers, "Visual-inertial navigation for a camera-equipped 25g nano-quadrotor," in *IROS2014 aerial open source robotics workshop*, 2014, p. 2.

[18] F. Ruffier, S. Viollet, S. Amic, and N. Franceschini, "Bio-inspired optical flow circuits for the visual guidance of micro air vehicles," *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03.*, vol. 3, pp. 846–849, 2003.

[19] W. E. Green and P. Y. Oh, "Optic-flow-based collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 15, no. 1, pp. 96–103, 2008.

[20] A. Briod, J.-c. Zufferey, and D. Floreano, "Optic-Flow Based Control of a 46g Quadrotor," *IEEE/RSJ International Conference on Robotics and Automation*, 2013.

[21] A. Briod, J.-C. Zufferey, and D. Floreano, "A method for ego-motion estimation in micro-hovering platforms flying in very cluttered environments," *Autonomous Robots*, vol. 40, no. 5, pp. 789–803, 2016.

[22] R. J. Moore, K. Dantu, G. L. Barrows, and R. Nagpal, "Autonomous mav guidance with a lightweight omnidirectional vision sensor," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3856–3861.

[23] T. Mori and S. Scherer, "First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1750–1757.

[24] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive avoidance using embedded stereo vision for mav flight," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 50–56.

[25] H. C. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 208, no. 1173, pp. 385–397, 1980.

[26] E. J. Smeur, Q. Chu, and G. C. de Croon, "Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 12, pp. 450–461, 2015.

[7] https://www.youtube.com/playlist?list=PL_KSX9GOn2P812tmddfrTlURHNieRe6YY